Does the latency matter?

Юрий Мусский Big Data Technologies





Yury Muski

SRE / DevOps at Big Data Technologies

Systems Engineer / DevOps 5 years

SysOps - 3 years







PageSpeed Insights

- ▲ Serve static assets with an efficient cache policy — 10 resources found
- Avoid chaining critical requests — 4 chains found
- Keep request counts low and transfer sizes small — 26 requests • 681 KB

Passed audits (21)

- Eliminate render-blocking resources — Potential savings of 40 ms
- Properly size images — Potential savings of 24 KB
- Defer offscreen images — Potential savings of 9 KB
- Minify CSS
- Minify JavaScript — Potential savings of 159 KB
- Remove unused CSS
- Efficiently encode images
- Serve images in next-gen formats — Potential savings of 16 KB
- Enable text compression
- Preconnect to required origins

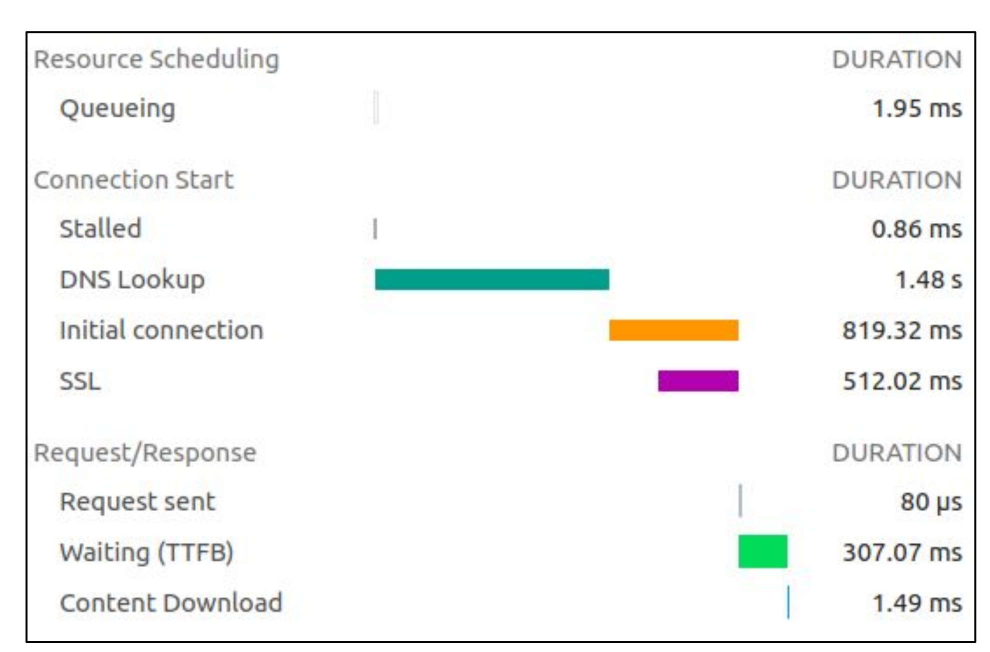
92

https://yurets.pro/

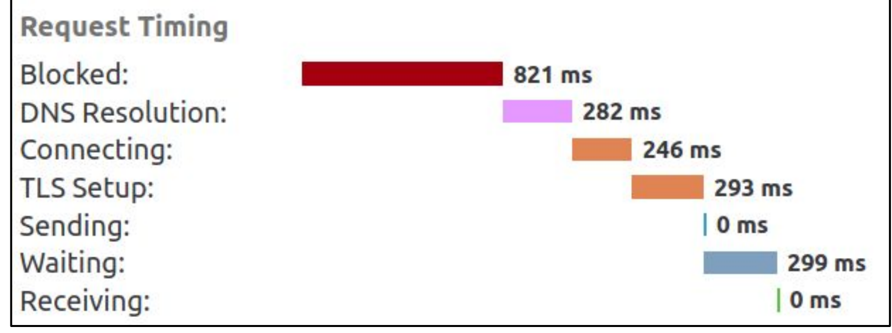


Browser Network Timings

Chrome



Firefox







Methodology:

DIG +trace

CURL -w

1000 requests

Rscript 99 percentile

HTTPSTAT visualization



Final Stats

99% latency ms

Setup/Location	EU region	Japan region
default	228	1102
tuned	122	554

Ratio 10x

(Tuned + GEO) or up to 1 second :)





DNS resolution

dnsperf.com

	DNS name	Query Speed
1	Sectigo	10.07 ms
2	Cloudflare	10.79 ms
3	DigitalOcean	11.5 ms
4	LimeLight DNS	14.67 ms
5	WordPress.com	14.96 ms
19	Route53	29.87 ms
28	Azure	40.65 ms
34	Google Cloud	56.92 ms
43	Afraid.org	124.39 ms

solvedns.com

Ranking	Name	Average
1	DNSMadeEasy	2.12
2	Verizon	2.8
3	NSOne	3.68
4	No-IP	4.52
5	CloudFlare	4.57
11	Route 53	23
16	Google	37.02
23	Afraid.org	115.11





Let's Test it =)

Provider	NS example	Domain
google	ns1.google.com	google.com
amazon	ns-81.awsdns-10.com	netflix.com
microsoft	ns1.msft.net	microsoft.com
cloudflare	ns3.cloudflare.com	cloudflare.com
hoster	ns1.hosterby.com	hoster.by
reg	ns1.reg.ru	reg.ru
afraid.org	ns7.afraid.org	freedns.afraid.org





```
yury@yury-laptop:~$
```

1

Start INTRO DNS TCP TLS HTTP Sum up End



DNS resolvers:

8.8.8.8 Google
9.9.9.9 IBM

1.1.1.1 Cloudflare
208.67.222.222 OpenDNS

Resolver/NS	**google**	**amazon**	**microsoft**	**cloudflare**	**hoster.by**	**reg.ru**	**afraid.org**
Google	54.00	12.59	24.00	7.51	92.00	44.00	780.87
Cloudflare	54.00	12.51	24.51	7.51	103.02	44.00	447.57
IBM	54.00	12.00	24.00	7.02	98.51	44.00	172.10
OpenDNS	54.00	13.00	24.00	6.51	80.51	45.02	423.06
Average (ms)	**54.00**	**12.53**	**24.13**	**7.14**	**93.51**	**44.26**	**455.90**





12ms aws VS 54ms google

44ms reg VS 93ms hoster

**Up to 450 ms delay
on afraid.org**

99% NS responce time (ms)

500.0
400.0
300.0
200.0
100.0
0.0

7.1
12.5
24.1
44.3
54.0
93.5
455.9

Average (ms)

cloudflare
amazon
microsoft
reg
google
hoster
afraid.org






DNS


Start
INTRO
DNS
TCP
TLS
HTTP
Sum up
End


HL
HighLoad++
Весна 2021

TCP (Connecting)

1 RTT (Round-trip time)

TCP Connection: SYN => SYN-ACK => ACK

Improvements:

- CDN Static
- CDN Dynamic
- Geo server distribution + Geo DNS



Due to #COVID—19, all TCP applications will be converted to UDP to avoid handshakes. 🤓





GEO DNS





Name	Type	Value	Geolocation	Set ID
demo.yurets.online.	A	34.84.69.230	*	default - jp
demo.yurets.online.	A	35.228.190.16	EU	europe

Routing Policy: Geolocation

Route 53 responds to queries based on the locations from which DNS queries originate.

Location: Default

Set ID: default - jp

Description of this record set that is unique within the group of geolocation sets.
Example:
Route to Seattle data center

Routing Policy: Geolocation

Route 53 responds to queries based on the locations from which DNS queries originate.

Location: Europe

Set ID: europe

Description of this record set that is unique within the group of geolocation sets.
Example:
Route to Seattle data center

TCP connection time depending on Geolocation

99% latency ms

Server / User location	**From server EU**	**From server JP**
EU response ms	36.6	263.9
Japan response ms	332.2	14.8
diff	**295.56**	**249.12**

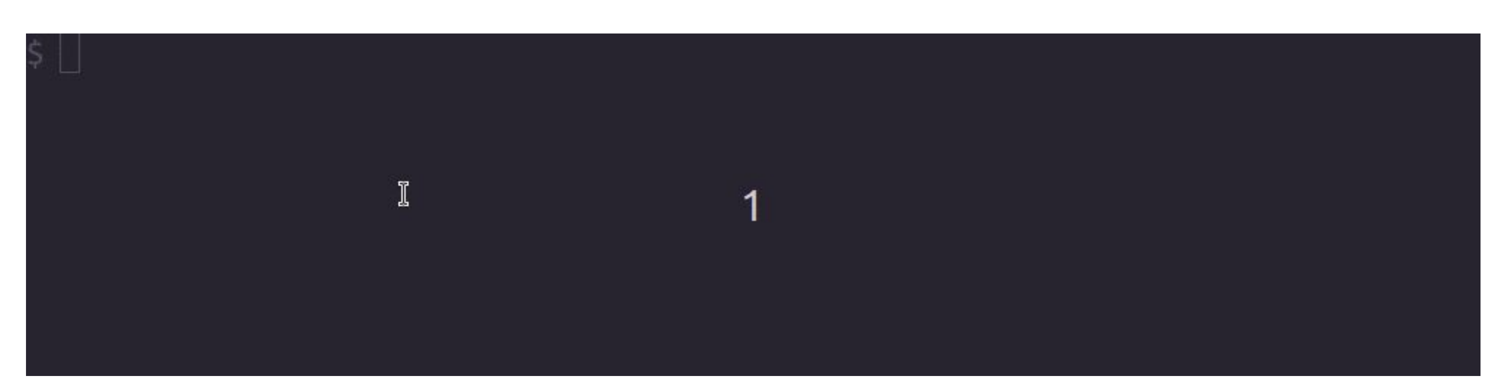
Geo DNS time savings up to 250-300ms per RTT





TCP Fast Open (TFO)

Checking on client:



Checking on server:

```
$ grep '^TcpExt:' /proc/net/netstat | cut -d ' ' -f 84-90  | column -t
TCPSYNChallenge  TCPFastOpenActive  TCPFastOpenActiveFail  TCPFastOpenPassive
0                0                  0                      3
```





TCP Fast Open (TFO)

Checking TFO on server:

cat /proc/sys/net/ipv4/tcp_fastopen

0 - disabled.
1 - only client (on outgoing connections)
2 - only server (on listening sockets)
3 - client + server

Enabling TFO:

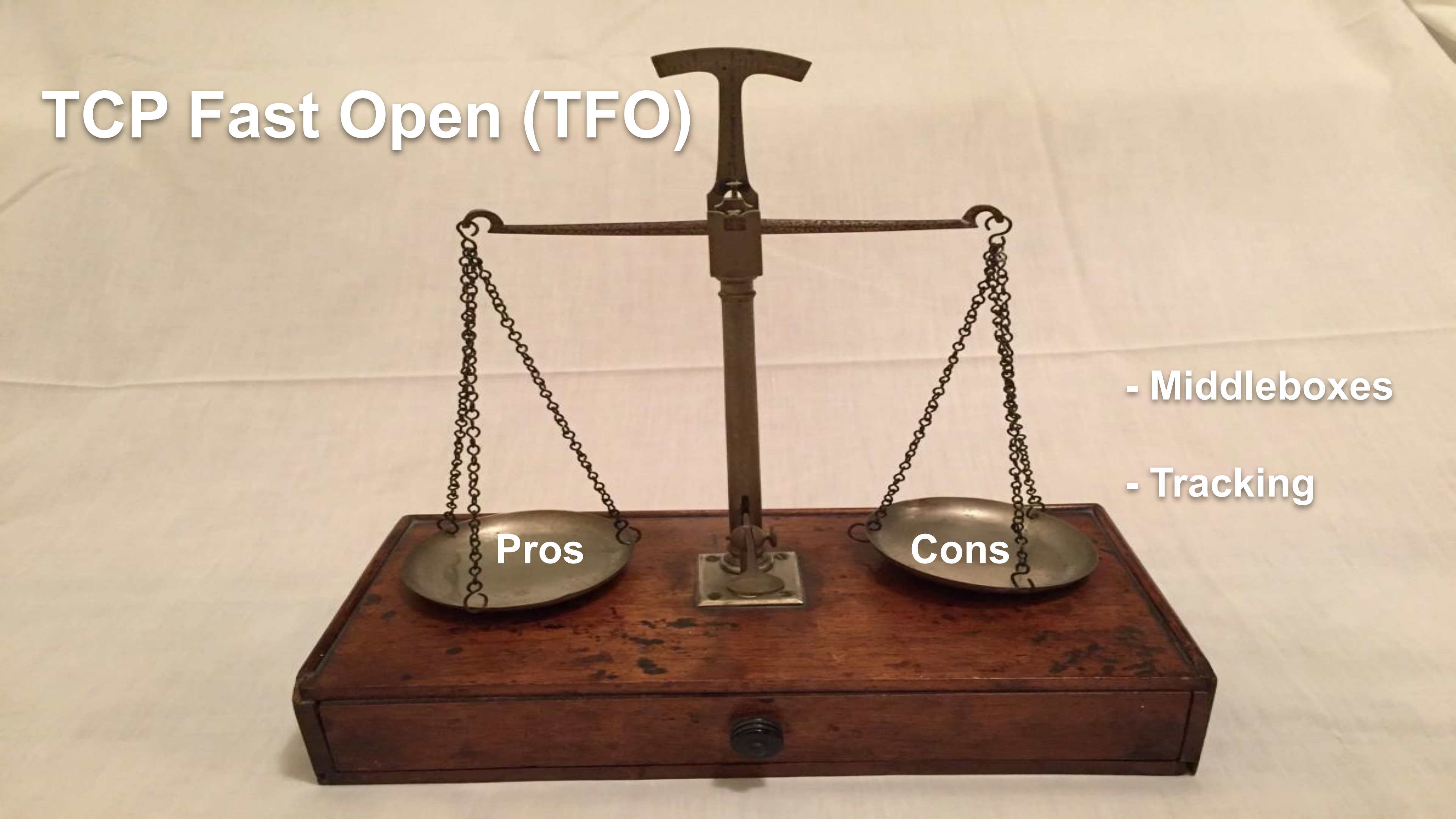
echo "3" > /proc/sys/net/ipv4/tcp_fastopen
or
echo "net.ipv4.tcp_fastopen=3" | sudo tee -a /etc/sysctl.conf
sudo sysctl -p /etc/sysctl.conf

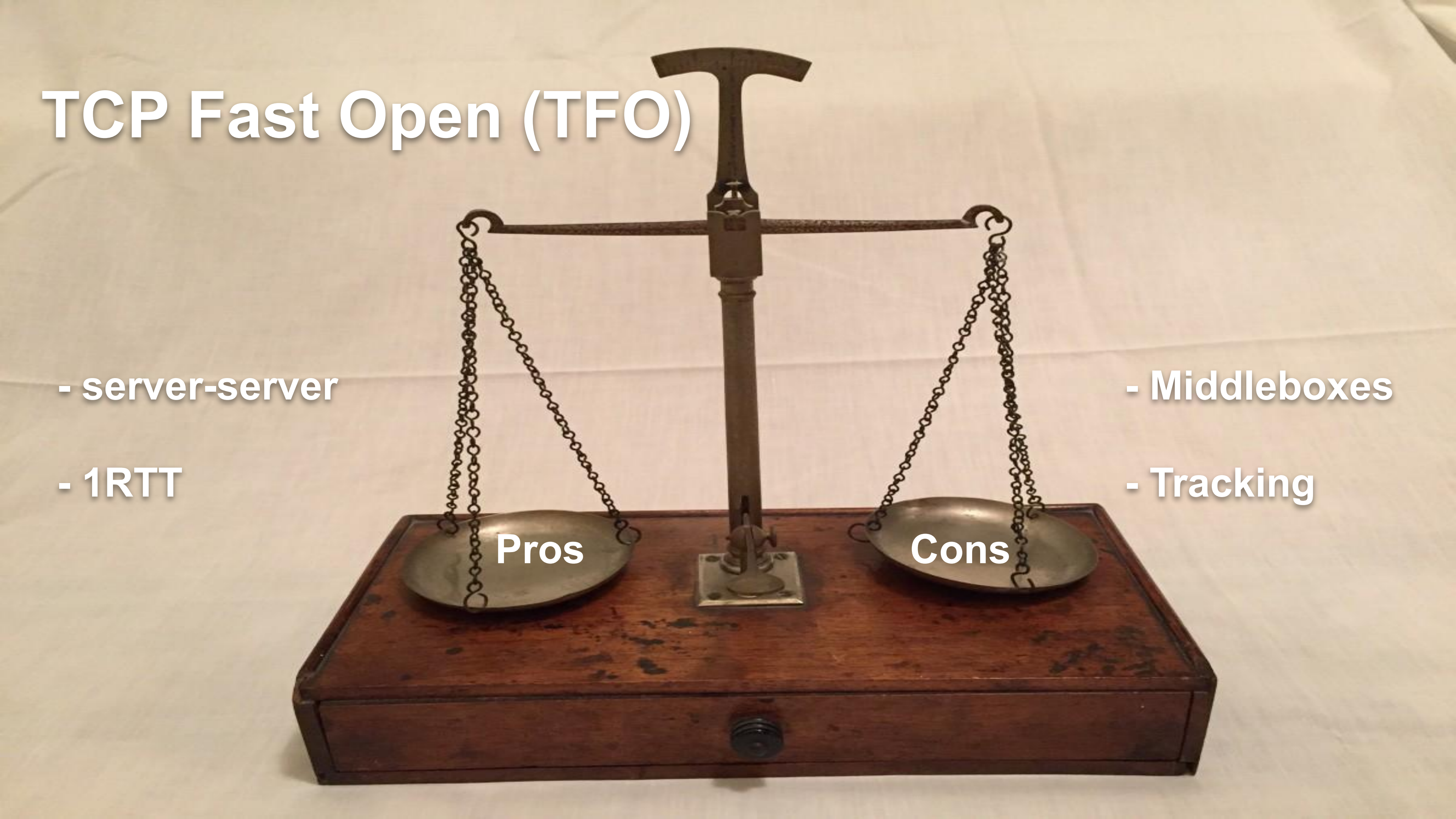
Adding fastopen to nginx config:

listen 80 fastopen=256

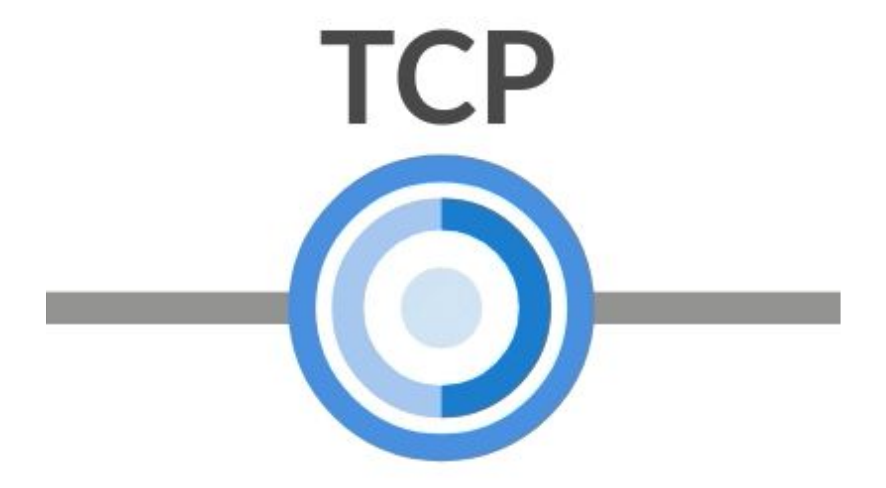





TCP Fast Open (TFO)
Pros
Cons
- Middleboxes
- Tracking


TCP Fast Open (TFO)
- server-server
- 1RTT
Pros
Cons
- Middleboxes
- Tracking

TCP




Start
INTRO
DNS
TCP
TLS
HTTP
Sum up
End


HL
HighLoad++
Весна 2021

TLS setup

0-2 RTT (Round-trip time)

Netscape Navigator 2.0

SSL v3 1996

TLS 1.1 2006

TLS 1.3 2018

1995 2000 2005 2010 2015 2020

TLS 1.0 1999

TLS 1.2 2008



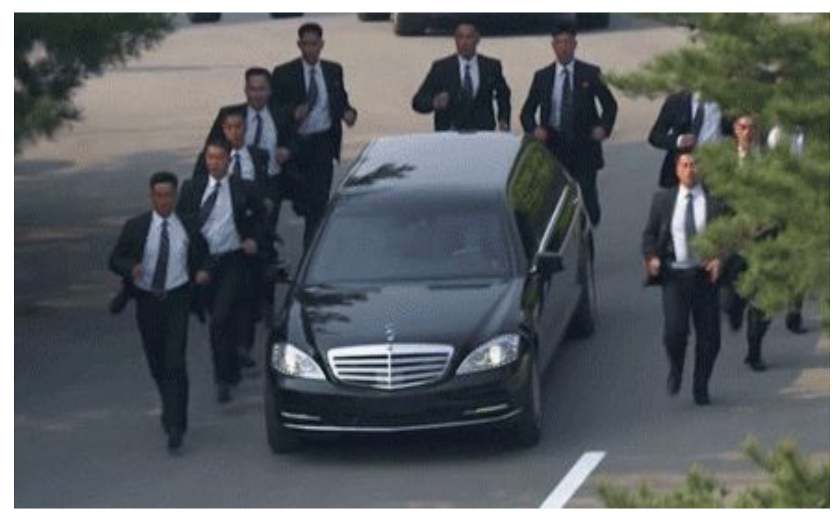
TLS 1.2 (2 RTT) vs TLS 1.3 (1RTT)

curl -v output:

```
TLSv1.3 (OUT), TLS handshake, Client hello (1):
TLSv1.3 (IN), TLS handshake, Server hello (2):
TLSv1.2 (IN), TLS handshake, Certificate (11):
TLSv1.2 (IN), TLS handshake, Server key exchange (12):
TLSv1.2 (IN), TLS handshake, Server finished (14):
TLSv1.2 (OUT), TLS handshake, Client key exchange (16):
TLSv1.2 (OUT), TLS change cipher, Change cipher spec (1):
TLSv1.2 (OUT), TLS handshake, Finished (20):
TLSv1.2 (IN), TLS handshake, Finished (20):
SSL connection using TLSv1.2 / ECDHE-RSA-AES256-GCM-SHA384
```

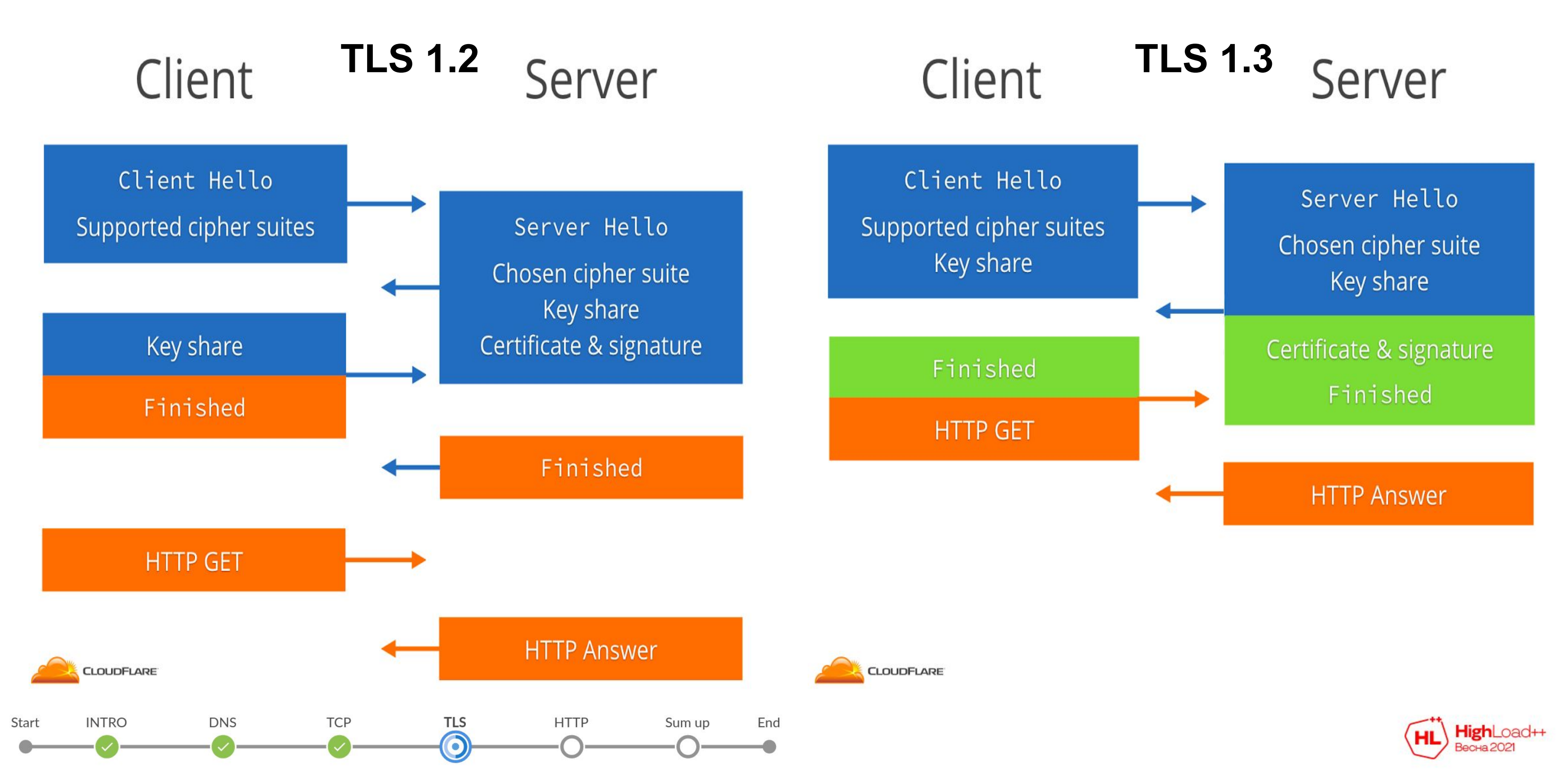
```
TLSv1.3 (OUT), TLS handshake, Client hello (1):
TLSv1.3 (IN), TLS handshake, Server hello (2):
TLSv1.3 (IN), TLS handshake, Encrypted Extensions (8):
TLSv1.3 (IN), TLS handshake, Certificate (11):
TLSv1.3 (IN), TLS handshake, CERT verify (15):
TLSv1.3 (IN), TLS handshake, Finished (20):
TLSv1.3 (OUT), TLS change cipher, Change cipher spec (1):
TLSv1.3 (OUT), TLS handshake, Finished (20):
SSL connection using TLSv1.3 / TLS_AES_256_GCM_SHA384
```

What TLS looks like:





TLS Handshake



TLS 1.2 vs 1.3
HTTPSTAT

$

1





TLS 1.2 vs TLS 1.3

99% latency ms

Server / User location	From server RU	From server US
TLS 1.2 JP response ms	550	267
TLS 1.3 JP response ms	280	136
diff	**270**	**131**
ratio	**196.4%**	**196.3%**

1 RTT benefit





TLS 1.3 0-rtt

TLS 1.2

Client
Server
1
2
3
4
5
6
7
300ms

TLS 1.3

Client
Server
1
2
3
4
5
200ms

TLS 1.3 + 0-RTT

Client
Server
1
2
3
100ms



TLS 1.3 0-rtt

nginx > 1.15.4, OpenSSL 1.1.1 or higher or BoringSSL

```
ssl_protocols TLSv1.3;
ssl_early_data on;
proxy_set_header Early-Data $ssl_early_data;
```





TLS 1.3 0-rtt

Checking:

```
host=tls13-0rtt.yurets.online # replace with your server name
echo -e "HEAD / HTTP/1.1\r\nHost: $host\r\nConnection: close\r\n\r\n" > request.txt
openssl s_client -connect $host:443 -tls1_3 -sess_out session.pem -ign_eof < request.txt
openssl s_client -connect $host:443 -tls1_3 -sess_in session.pem -early_data request.txt
```

```
Early data was accepted
Verify return code: 0 (ok)
---
HTTP/1.1 200 OK
```





RSA key length

RSA Decryption time by key length

Decryption time (milliseconds): 0, 200, 400, 600, 800, 1000, 1200

Key (modulus) length, bits: 512, 768, 1024, 1280, 1536, 1792, 2048, 2304, 2560, 2816, 3072, 3328, 3584, 3840, 4096

With every doubling of the RSA key length, decryption is 6-7 times slower.





TLS config best practice



Server Software

- ( ) Apache
- ( ) AWS ALB
- ( ) AWS ELB
- ( ) Caddy
- ( ) Dovecot
- ( ) Exim
- ( ) Golang
- ( ) HAProxy
- ( ) lighttpd
- ( ) MySQL
- (•) nginx
- ( ) Oracle HTTP
- ( ) Postfix
- ( ) PostgreSQL
- ( ) ProFTPD
- ( ) Tomcat
- ( ) Traefik

Mozilla Configuration

- (•) Modern
  Services with clients that support TLS 1.3 and don't need backward compatibility
- ( ) Intermediate
  General-purpose servers with a variety of clients, recommended for almost all systems
- ( ) Old
  Compatible with a number of very old clients, and should be used only as a last resort

Environment

Server Version	1.16.1
OpenSSL Version	1.1.1

Miscellaneous

- [x] HTTP Strict Transport Security
  This also redirects to HTTPS, if possible
- [x] OCSP Stapling

https://ssl-config.mozilla.org/






TLS


Start
INTRO
DNS
TCP
TLS
HTTP
Sum up
End


HL
HighLoad++
Весна 2021

HTTP (Sending-Waiting-Receiving)

1 RTT (REQUEST=>RESPONSE)

Year	HTTP Version
1997	1.1
2015	2.0
2019	3.0







HTTP/2

Client Server Client Server Client Server

GET index.html
GET style.css
GET query.js

HTTP/1.1

GET index.html
GET style.css
GET query.js

HTTP/2 (no Push)

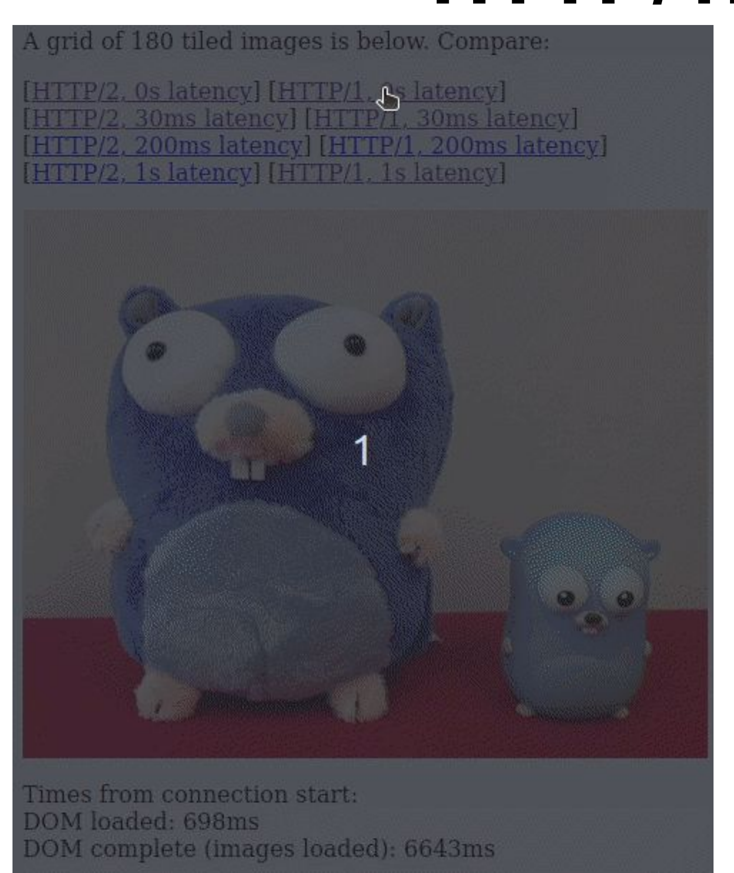
GET index.html

HTTP/2 (with Push)

Request Response Resources



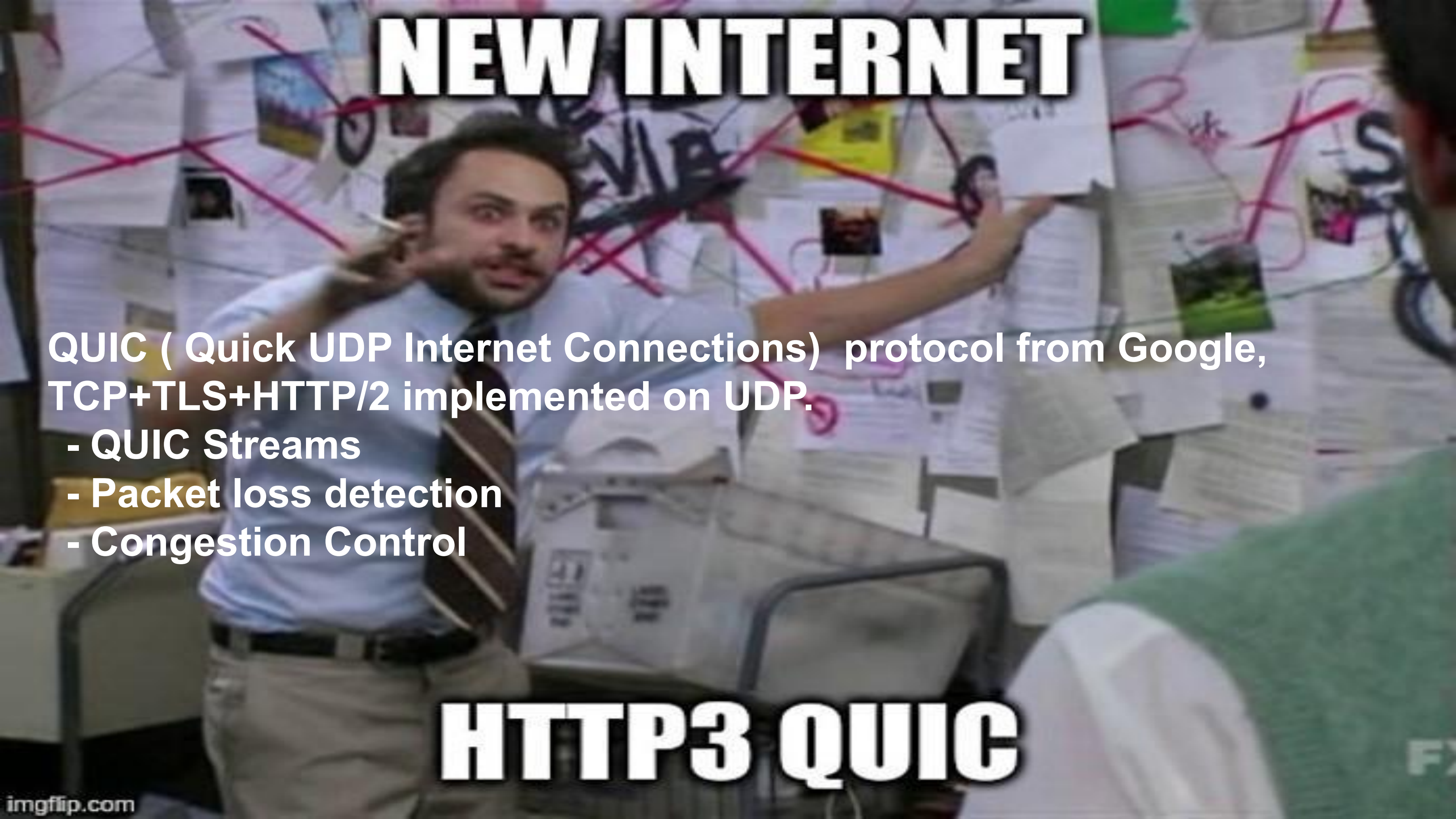
HTTP/1.1 vs HTTP/2



Test it:






NEW INTERNET
QUIC (Quick UDP Internet Connections) protocol from Google,
TCP+TLS+HTTP/2 implemented on UDP.
- QUIC Streams
- Packet loss detection
- Congestion Control
HTTP3 QUIC
imgflip.com

WHAT DO WE WANT?

HTTP3 SUPPORT





Enable HTTP/3

Compile nginx manual:
https://github.com/cloudflare/quiche/tree/master/extras/nginx#readme

docker image: ymuski/nginx-quic

Nginx config:

listen 443 quic reuseport;
add_header alt-svc 'h3-29=":443"; ma=86400';



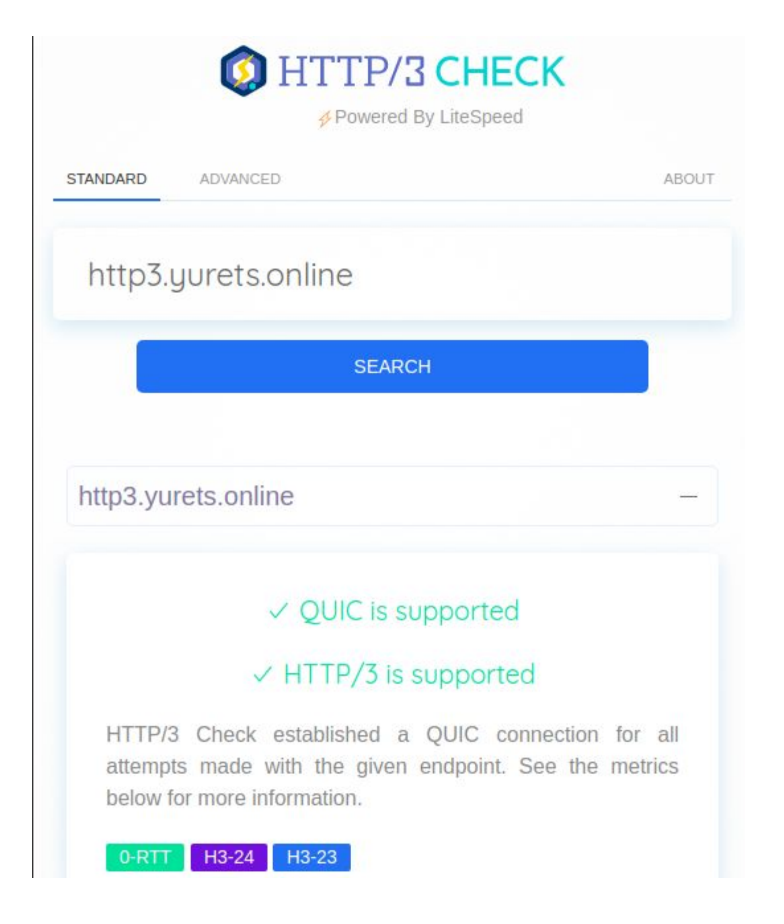



PSST!
OPEN 443/UDP ON FIREWALL
imgflip.com

Test HTTP/3

Test online:
https://www.http3check.net/









Test HTTP/3

Compile curl manual:
https://github.com/curl/curl/blob/master/docs/HTTP3.md

docker image: ymuski/curl-http3

docker run -it --rm ymuski/curl-http3 curl -Lv https://http3.yurets.online **--http3**

nginx log:

```
13.48.179.147 - - [19/Feb/2020:13:47:48 +0000] "GET /hello HTTP/3" 200 12 "-" "curl/7.69.0-DEV"
46.53.240.56 - - [19/Feb/2020:13:47:48 +0000] "GET /hello HTTP/3" 200 12 "-" "curl/7.69.0-DEV"
```

```
$ docker run -it --rm ymuski/curl-http3 curl -Lv https://http3.yurets.online --http3
*   Trying 35.187.196.211:443...
* Sent QUIC client Initial, ALPN: h3-25h3-24h3-23
* h3 [:method: GET]
* h3 [:path: /]
* h3 [:scheme: https]
* h3 [:authority: http3.yurets.online]
* h3 [user-agent: curl/7.69.0-DEV]
* h3 [accept: */*]
* Using HTTP/3 Stream ID: 0 (easy handle 0x558482439780)
> GET / HTTP/3
> Host: http3.yurets.online
> user-agent: curl/7.69.0-DEV
> accept: */*
>
< HTTP/3 200
< server: nginx/1.16.1
< date: Wed, 19 Feb 2020 14:05:46 GMT
< content-type: text/html
< content-length: 12
< last-modified: Sun, 16 Feb 2020 15:53:01 GMT
< etag: "5e49655d-c"
< alt-svc: h3-24=":443"; ma=86400, h3-23=":443"; ma=86400
< accept-ranges: bytes
<
Hello HTTP3
```

Browsers and HTTP/3

Chrome	Stable build (89)	May 2021	#enable-quic
Firefox	Stable build (88)	May 2021	network.http.http3.enabled

Request URL: https://http3.yurets.online/hello
Request Method: GET
Remote Address: 34.85.47.11:443
Status Code: 200 OK
Version: HTTP/3

Filter Headers

Response Headers (179 B)

alt-svc: h3-24=":443"; ma=86400, h3-23=":443"; ma=86400

https://developers.cloudflare.com/http3/





HTTP/2 vs HTTP/3

99% latency

HTTP Protocol/User location	From server RU	From server US
HTTP2 JP response ms	828	419
HTTP3 JP response ms	552	368
ratio	**1.5**	**1.14**

HTTP/3 response is 1.14x-1.5x faster than HTTP/2.





HTTP/3 + DNS

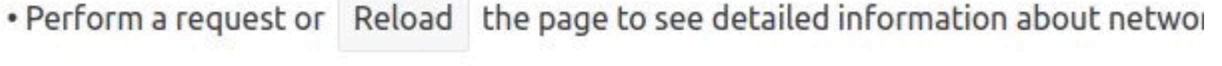
Type	Name	TTL
HTTPS	yurets.pro	Auto

Priority	Target	Value
1 (0 - 65535)	yurets.pro.	alpn=""h3,h2""

Search with Google or enter address

Inspector | Console | Debugger | Network | Style Editor

Filter URLs

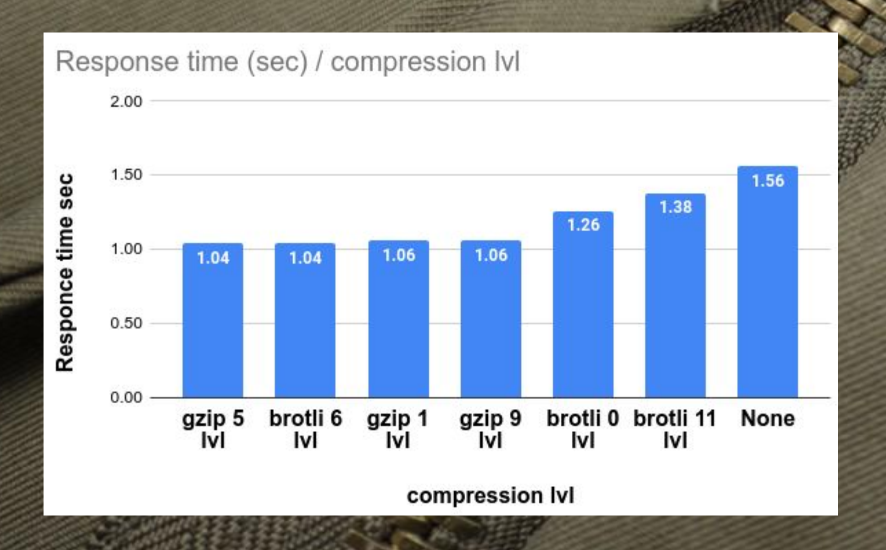




HTTP Compression

Less response size => Faster transfer
Gzip 1-9 lvls
Brotli 0-11 lvls

Json 137kb file check:





HTTP Cache

use cache =)

Etag and Last-modified headers - weak caching headers (validators)

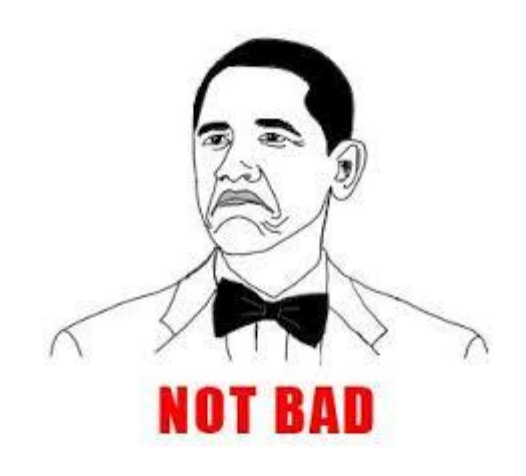
Expires and Cache-control - strong caching headers (refresh information)





Sum up

Use fast NS server	~ 50ms
Geo location or CDN	~ 300 ms per RTT
TFO if suits	1 RTT
TLS 1.3	1 RTT
Early data if suits	1 RTT
HTTP2	multiple req/resp in parallel
Cache + Compression	just use it
Try HTTP 3	possible 10-50% benefit no 1st redirect delay







Useful links:

site:
yurets.pro



repo:
github.com/yurymuski/demo-latency



